EXTRACTING DEEP NEURAL NETWORK BOTTLENECK FEATURES USING LOW-RANK MATRIX FACTORIZATION

*Yu Zhang, Ekapol Chuangsuwanich, James Glass**

MIT Computer Science and Artificial Intelligence Laboratory,
Cambridge, Massachusetts 02139, USA

{yzhang87, ekapolc, glass}@mit.edu

ABSTRACT

In this paper, we investigate the use of deep neural networks (DNNs) to generate a stacked bottleneck (SBN) feature representation for low-resource speech recognition. We examine different SBN extraction architectures, and incorporate low-rank matrix factorization in the final weight layer. Experiments on several low-resource languages demonstrate the effectiveness of the SBN configurations when compared to state-of-the-art hybrid DNN approaches.

Index Terms— DNN, Bottleneck features

1. INTRODUCTION

Currently, there are two main approaches for incorporating neural networks into hidden Markov models (HMM) for automatic speech recognition (ASR). The *hybrid* approach uses a network to directly estimate HMM emission probabilities, while the *tandem* approach uses a network to produce a feature vector that can be modeled by Gaussian mixture models (GMMs). This is done by training a network to predict phonetic targets, and then either using the estimated target probabilities [1] or the activations of a narrow hidden "bottleneck" (BN) layer [2] as features for a standard GMM-HMM configuration. Recently, deep neural networks (DNNs) have been used with great success in hybrid systems, achieving excellent results on ASR tasks [3, 4, 5, 6]. While BN features have been successfully used for ASR for many years, there has been less work examining the use of DNNs for BN feature extraction.

In 2011, Yu and Seltzer applied a DNN for extracting BN features, with the bottleneck being a small hidden layer placed in the middle of the network [7]. They found that this method for generating BN features increased ASR accuracy when using context-dependent (CD) targets for training. Currently however, tandem frameworks using BN features do not perform as well as the best DNN hybrid systems. Some have argued that by placing a BN layer in the middle of the DNN, the frame accuracy of the output targets degrades [8, 9]. Therefore, BN features trained this way cannot achieve the full benefits of DNNs. To circumvent this, Sainath *et al.* trained the DNN without a BN layer. Then, to reduce the dimensionality, an auto-encoder neural network was trained on the MLP outputs with a BN layer [8]. Alternatively, Yan *et al.* trained a DNN and used PCA to do dimensionality reduction on the last hidden layer [9]. There has also been work in another direction for improving BN features. In [10], researchers at Brno University of Technology started using a stacked bottleneck (SBN) architecture and linear activation function. In [11], they investigated the SBN architecture using DNN architectures. However, they used context-independent (CI) target outputs, and did not compare the results with hybrid systems.

In [12], Sainath *et al.* proposed a low-rank matrix factorization method for the final weight layer of hybrid DNN systems and demonstrated no significant loss in ASR accuracy compared to the ones without. In this work, we consider this approach and apply it to BN feature extraction. Applying this approach to BN feature extraction has two benefits. First, the low-rank matrix factorization ensures that there is no significant loss in terms of cross entropy during training. Second, directly extracting BN features from the low-rank layer allows for more information to be encoded, since there is no non-linear sigmoid compression between the BN features and the activation output. In this research, we combine this low-rank framework and the SBN configuration of [11] to improve BN feature extraction for tandem ASR. Results on several languages from the Babel project show over 10% relative gains over standard PLP features, and comparable results to hybrid DNN systems while having much shorter training time.

The rest of this paper is organized as follows. In Section 2, we present the details of our new approach. We describe our experimental setup in Section 3. In Section 4, we report our experimental results and analyze the performance of different architectures. Finally, we conclude the paper in Section 5.

*Supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense US Army Research Laboratory contract number W911NF-12-C-0013. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

2. MODEL DESCRIPTION

Before we describe the proposed method, we provide an overview of the two related efforts that inspired our BN architecture: the low-rank matrix factorization for DNN weights, and the SBN framework.

2.1. Low-rank matrix factorization

The left side of Figure 1 shows a typical ASR DNN architecture. Following Sainath *et al.*, [12] we investigate a low-rank approximation to the weights of the softmax layer of the network. By considering the weights of the softmax layer as a matrix, we can factorize the weight matrix into two matrices of lower rank. As illustrated by the right side of Figure 1, this is done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units followed by a softmax layer. More specifically, a new BN output layer with r linear hidden units is inserted into the last weight matrix with a hidden layer of size h, and a softmax layer with s state posterior outputs. This changes the number of parameters from $h * s$ to $r * (h + s)$. Notice that there is no non-linearity for this BN output layer. Instead of using this structure for hybrid DNNs, we use it for extracting BN features. There are two benefits of using this method. First, it ensures the best achievable frame accuracy even with a relatively small r. Second, the linearity of the output for the BN layer prevents any loss of information when we treat the DNN as a feature extractor.

2.2. Stacked bottleneck (SBN) features

The idea of using hierarchical processing of neural networks (NNs) has been explored by several researchers. Valente *et al.* uses a second NN to help correct the posterior outputs of the first NN by feeding it a different set of features [13]. In the context of the Babel project, SBN features have shown promising results in [11]. One argument for the usage of these cascading structures is that they enable more information, such as additional context, to be utilized more effectively [14].

2.3. Low-Rank Stacked Bottleneck (LrSBN)

Figure 2 gives an overview of our proposed low-rank SBN feature extraction framework. The BN outputs from the first DNN are concatenated with context expansion and fed to the second DNN. This structure is similar to [11] except we always place the linear BN layer (for the low-rank factorization) in the *last* hidden layer instead of a sigmoid BN layer in the middle of the network. Experiments in [12] have shown that the hidden layers do not have the same low-rank properties as the weights in the softmax layer. We also use tied-states as the output targets instead of CI targets.

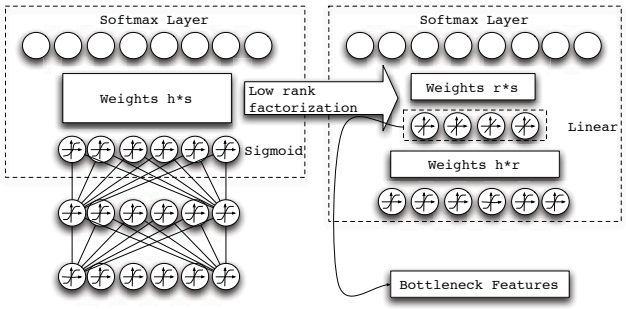

Fig. 1. Diagram of the low-rank factorized DNN.

3. SYSTEM DESCRIPTION

3.1. Data

In these experiments, we use data from the Babel project, which consists of a growing collection of resources to facilitate ASR and keyword spotting research on low-resource languages. Each language consists of two speaking styles: scripted (prompted speech) and conversational (spontaneous telephone conversations). There are also currently two standard training conditions: Full ($\sim$80 hours) and Limited ($\sim$10 hours). In our experiments, we only use the conversational data for training. The standard 10-hour Dev sets containing only the conversational data are used for testing. In this paper, we report on the Turkish, Assamese, and Bengali language packs from releases IARPA-babel105b-v0.4, IARPA-babel102b-v0.4, and IARPA-babel103b-v0.3, respectively.

3.2. Baseline HMM systems

Our baseline HMM systems were trained using the Kaldi ASR toolkit [15]. We used 13-dimensional PLP features concatenated with F0 estimates and the probability of voicing [16]. Conversation-based mean and variance normalization was applied in both training and testing stages. The resulting 15-dimensional features were concatenated using ± 4 frames before and after the middle frame resulting in 135-dimensional vectors. LDA and MLLT [17] were applied to reduce the dimensionality and orthogonalize the features. Finally, a global fMLLR [18] was applied to normalize inter-speaker variability. For acoustic modeling, we used phonetic decision-based tied-state triphone CD-HMMs with $\sim$2500 states and 18 Gaussian components per state. Trigram language models were created from training data transcripts.

3.3. Baseline hybrid DNN systems

The hybrid DNN systems were also created using Kaldi [15]. The DNNs had 6 hidden layers. The output layer was a softmax layer with target outputs corresponding to CD-HMM states. The network inputs were the speaker adapted features from the CD-HMM baseline (both for training and test) and concatenated using ± 5 frames (the total size is $40 * 11 = 440$).

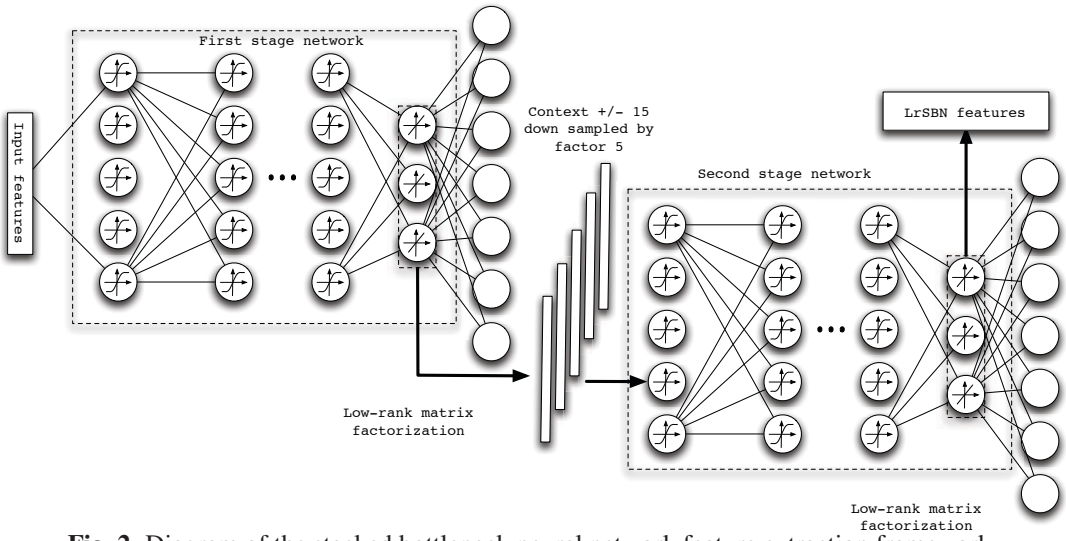

Fig. 2. Diagram of the stacked bottleneck neural network feature extraction framework.

We used 1024 hidden units for each hidden layer. The nonlinearities in the hidden layers were sigmoid functions, and the objective function is the cross-entropy criterion. The alignment of CD states to frames was derived from the CD-HMM baseline systems and remained fixed during training.

The DNN was pre-trained by restricted Boltzmann machines. The initialization of the network, the fine-tuning, and the learning rate followed the setting in [19]. We also perform sequence training [20].

3.4. Low-Rank stacked bottleneck (LrSBN) systems

The input features for the first DNN (Figure 2) follows the method of [11]. 23 critical-band energies are obtained from a Mel filter-bank, with conversation-side-based mean subtraction. These features are augmented with pitch and probability of voicing. 11 consecutive frames are stacked together. Each of the 23+2 dimensions is then multiplied by a Hamming window across time, and a DCT is applied for dimensionality reduction. The 0th to 5th coefficients are retained, resulting in a feature of dimensionality $(23 + 2) * 6 = 150$.

The input features of the second DNN are the outputs of the BN layer from the first DNN. Context expansion is done by concatenating frames with time offsets $-10, -5, 0, 5, 10$. Thus, the overall time context seen by the second DNN is 31 frames. Both DNNs use same setup of 5 hidden sigmoid layers and 1 linear BN layer, and both use tied-states as target outputs. The targets are generated by forced alignment from the HMM baseline. No pre-training is used. Finally, the raw BN outputs from the second DNN are whitened using a global PCA and used as features for a conventional CD-HMM system.

4. ANALYSIS OF LRSBN FEATURES

4.1. Context-independent vs. context-dependent labels

In [11], a DNN was trained to classify CI states. However, we have found, as have others [7], that using CD targets produces better results. Table 1 compares word error rates (WERs) between BN systems trained from CI versus CD labels on the Turkish limited condition task. The PLP-based baseline for this task had a WER of 75.0%. The first column in the table shows that if CD labels are used to train a single stage network, a 1.2% WER gain is obtained over CI labels. A gain of 2.2% is obtained when CD labels are used to train the stacked network. Therefore, in all subsequent experiments described here, we use only CD labels for SBN training.

Table 1. Turkish BN WERs for CI vs CD label training.

	Single DNN	Stacked DNN
CI targets	69.6%	68.8%
CD targets	68.4%	66.6%

4.2. The best layer for bottleneck placement

Past research [12] has shown that DNN hidden layers do not all have the same low-rank properties. Following this observation, we compare the cross entropy per frame for different DNN configurations. Table 2 shows the average cross entropy (CE) per frame on the cross validation set for different BN placements on the Bengali Limited condition task. In all setups, the BN layers have 80 hidden units. As the table shows, putting a low-rank linear layer in the middle performs worse than a typical sigmoid BN layer. On the other hand, the low-rank softmax layer also has the lowest cross entropy

per-frame. Thus, for all remaining experiments, we put the BN layer as the last hidden layer.

Table 2. DNN Comparisons of average CE per frame. 'Last' refers to putting the BN layer right before the softmax layer.

BN Type	Sigmoid layer		Low-rank linear layer	
BN Location	Middle	Last	Middle	Last
Avg. CE	0.253	0.250	0.257	**0.245**

4.3. Low-rank on the softmax layer

In order to determine whether the low-rank factorization on the softmax layer is necessary, we also evaluated the features generated by different activation functions on the Bengali Limited condition. The PLP-based GMM-HMM baseline for this task achieved 75.3% WER. In Table 3, we compare the results we achieve with BN features using a standard sigmoid on the softmax layer (SBN) with those obtained using the low-rank formulation (LrSBN). We also consider two BN derivations. In the first row of Table 3, we use the BN feature directly without any post-processing. In the second row, we apply PCA to raw BN features and reduce the dimensionality from 80 to 30. We then add Δ and Δ^2 BN features to model additional contextual information. It can be seen that for both conditions, the LrSBN achieves better performance.

Table 3. Sigmoid vs. low-rank, and BN feature comparison.

	SBN	LrSBN
raw BN	70.8%	69.2%
raw BN (PCA) + Δ + Δ^2	68.1%	67.2%

4.4. Results on larger tasks and different languages

Further evaluation of the proposed method was performed on different languages with a speaker-adapted model. On this task, we compared the baseline GMM-HMM system, the hybrid DNN-HMM system, and the LrSBN system. The top of Table 4 shows that with standard ML training, an improvement of over 10% relative could be achieved when using LrSBNs. This result is even better than a hybrid DNN system that uses speaker-adapted input features. After speaker-adaptated training (SAT) and minimum Bayes risk (MBR) discriminative training [21] on the LrSBN features, the performance of the LrSBN system is similar to the hybrid DNN system with sequence training (SQ) in Bengali, while performing 0.6% better in the Assamese case.

In addition to examining the limited condition training task, we also quantified the performance of the LrSBN features on the Bengali Full condition task. The WER comparisons are shown in Table 5. It can be seen the gain compared to the PLP baseline is even larger than for the limited condition case, with a 12.6% relative gain. It is also better than the Hybrid DNN system, improving the performance from 59.4% to 56.4%. Compared to the hybrid system with sequence training, the performance is still slightly better. Note that using sequence training has its disadvantages in terms of training time. For example, using a Tesla K20 GPU, an iteration of sequence training took up to 6 hours compared to the 40 minutes for cross entropy training. This can be an important constraint for the Babel project which emphasizes rapid system deployment. Using BN features also better lends itself to further improvements using standard techniques. Such improvements include better use of context information via fMPE [22], and using speaker-adapted features as DNN inputs. We plan to explore these as future work.

Table 4. Results on Bengali and Assamese Limited tasks.

AM Training	ML	LDA+MLLT	SAT+MBR
Bengali Limited Condition			
PLP+F0	75.3%	74.4%	71.8%
DNN-HMM	n/a	n/a	68.0%
DNN-HMM+SQ	n/a	n/a	66.1%
LrSBN+Δ+Δ^2	67.2%	n/a	**66.0%**
Assamese Limited Condition			
PLP+F0	74.6%	73.0%	70.5%
DNN-HMM	n/a	n/a	67.2%
DNN-HMM+SQ	n/a	n/a	65.7%
LrSBN+Δ+Δ^2	65.8%	n/a	**65.2%**

Table 5. WER comparisons for the Full Bengali task.

AM Training	ML	LDA+MLLT	SAT+MBR
PLP+F0	69.2%	68.4%	64.5%
DNN-HMM	n/a	n/a	59.4%
DNN-HMM+SQ	n/a	n/a	55.6%
LrSBN+Δ+Δ^2	59.6%	n/a	**55.4%**

5. SUMMARY AND FUTURE WORK

In this paper, we explored low-rank matrix factorization of the final weight layer for extracting stacked bottleneck features. The results on multiple languages and training conditions demonstrated that the LrSBN features could achieve WERs similar to, or in some cases superior to, the performance of state-of-the-art hybrid DNN systems while being significantly faster to train. Our ongoing work include the investigation of unsupervised training in the proposed approach and building multi-lingual system using the LrSBN features.

6. REFERENCES

[1] H. Hermansky, D. P. W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Proc. ICASSP*, 2000, pp. 1635–1639.

[2] F. Grézl, M. Karafiát, S. Kontár, and J. Černocký, "Probabilistic and bottle-neck features for LVCSR of meetings," in *Proc. ICASSP*, 2007, vol. 4, pp. 757–761.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," in *IEEE Trans. on Audio, Speech, and Language Processing*, 2012, vol. 20, pp. 30–42.

[4] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. InterSpeech*, 2011, pp. 437–440.

[5] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum bayes risk training of deep neural network acoustic models using distributed hessian-free optimization," in *Proc. InterSpeech*, 2012.

[6] N. Jaitly, P. Nguyen, A. Senior, and V. Vanhoucke, "Application of pretrained deep neural networks to large vocabulary speech recognition," in *Proc. InterSpeech*, 2012.

[7] D. Yu and M. L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," in *Proc. InterSpeech*, 2011, pp. 273–240.

[8] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Proc. ICASSP*, 2012, pp. 4153–4156.

[9] Z. J. Yan, Q. Huo, and J. Xu, "A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR," in *Proc. InterSpeech*, 2013.

[10] K. Veselý, M. Karafiát, and F. Grézl, "Convolutive bottleneck network features for LVCSR," in *Proc. ASRU*, 2011, pp. 42–47.

[11] M. Karafiát, F. Grézl, M. Hannemann, K Veselý, and J. H. Černocký, "BUT babel system for spontaneous cantonese," in *Proc. InterSpeech*, 2013.

[12] T. N.Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, 2013.

[13] F. Valente, J. Vepa, C.Plahl, et al., "Hierarchical neural networks feature extraction for LVCSR system," in *Proc. InterSpeech*, 2007.

[14] M. Karafiát and F. Grézl, "Hierarchical neural net architectures for feature extraction in ASR," in *Proc. InterSpeech*, 2010.

[15] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz, J. Silovský, G. Stemmer, and K. Veselý, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[16] D. Talkin, *A Robust Algorithm for Pitch Tracking*, chapter 4, Speech Coding and Synthesis, 2013.

[17] M. J. F. Gales, "Semi-tied covariance matrices for hidden markov models," in *IEEE Trans. on Audio, Speech, and Language Processing*, 1999.

[18] M. J. F. Gales, "Maximum likelihood linear transformation for HMM-based speech recognition," in *Comp.Speech & Language*, 1998.

[19] S. P. Rath, D. Povey, K. Veselý, and J. H. Černocký, "Improved feature processing for deep neural networks," in *Proc. InterSpeech*, 2013.

[20] K. Veselý, A. Ghoshal, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. InterSpeech*, 2013.

[21] M. Gibson and T. Hain, "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition," in *Proc. InterSpeech*, 2006, pp. 2406–2409.

[22] D. Povey, B. Kingsbury, L. Mangu, et al., "fMPE: Discriminatively trained features for speech recognition," in *Proc. ICASSP*, 2005.